УДК 004.056.5:655.25

ЕЛЕМЕНТИ ЗАХИСТУ ЕТИКЕТКОВО-ПАКУВАЛЬНОЇ ПРОДУКЦІЇ

Бізюк А.В.
к.т.н., професор, кафедра Медіасистем та технологій
Харківський національний університет радіоелектроніки

Анотація. *Контроль поліграфічних захисних елементів обумовлює правильність їх виконання, зменшення кількості неякісного товару і підробок, що особливо актуально в сучасних умовах. Результатом дослідження є аналіз методів перевірки рівня захисту етикетково-пакувальної продукції, розроблений алгоритм перевірки зображень на збіг з оригіналом, а також відповідний програмний засіб. Дане дослідження розширює можливості ефективного контролювання захисних елементів на додрукарській стадії виробництва.*

Ключові слова: *ЕЛЕМЕНТИ ПОЛІГРАФІЧНОГО ЗАХИСТУ, АЛГОРИТМ, ПОРІВНЯННЯ, ФАЛЬСИФІКАТ.*

Вступ

Фальсифікація – це створення зразка продукції, максимально близького до оригіналу, для підміни останнього з метою отримання прибутку [1]. Розвиток копіювальної техніки приніс нові можливості створення підробних екземплярів поліграфічної продукції. В сучасному світі покращуються й можливості фальсифікаторів: кожен новий спосіб друку або спрощує створення підробки, або примушує виготівників фальсифікованої продукції обходити цей спосіб за допомогою інших, тим самим, удосконалюючи процес друку.

Проблема фальсифікації полягає у фінансовому і моральному збитку виробника і споживача, особливо важливою є шкода, що наноситься здоров'ю людини. Говорячи про етикетково-пакувальну продукцію, слід зазначити, що поліграфічна підробка є вторинною відносно підробки самого продукту, проте захисну функцію несе саме вона.

Ринок етикетково-пакувальної продукції дуже привабливий для фальсифікаторів. Це обумовлено великими і незмінно зростаючими обсягами виробництва і прибутку. У 2019 році світовий ринок друку упаковки і етикетки показав зростання у розмірі 11,4% в порівнянні з 2018 роком [2]. Статистика показує тенденцію зростання ринку і останніми роками. За даними досліджень Smithers Pira, в 2016 році прибуток ринку упаковки складав 820 мільярдів доларів [3]. Середній об'єм доходу від етикеткової промисловості складає близько 15 мільярдів доларів на рік, і також продовжує рости.

Це зростання забезпечене рядом тенденцій, таких як зростаюча урбанізація, інвестиції в будівництво, розширення сфери охорони здоров'я і розвиток економіки країн, у тому числі, Китаю, Індії, Бразилії і деяких країн Східної Європи. Збільшення особистого доходу і зміна способу життя в таких

країнах впливає на споживання продукції і, побічно, на ринок етикетково-пакувальної продукції. Підвищений попит на побутову техніку, наприклад, на пральні машини, викликає не лише зростання попиту на упаковку для самих машин, але і на потребі в пов'язаній продукції, такий як, засоби догляду за побутовою технікою і засоби, вживані разом з нею, наприклад, пральні порошки. У свою чергу для супровідних товарів також потрібні етикетки і упаковки. Високі темпи зростання споживання етикетково-пакувальної продукції наявні в таких областях виробництва, як косметика, засоби побутової і особистої гігієни, одяг, аудіо і відео продукція, техніка, і, особливо, продукти харчування, електротовари, фармацевтичні і медичні вироби, що стимулює виробництво упаковок і етикеток.

При цьому об'єм контрафактної продукції оцінюється більш ніж в 20% усієї продукції світового ринку [5]. Виробники всього світу щодня стикаються з проблемою підробки продукції. Окрім прямих збитків, компанії-виробники також несуть втрати від судових витрат, які є присутній при виявленні підробки споживачем.

Таким чином, ситуація, що склалася на ринку, показує, що особливо важливо приділяти увагу захисним функціям етикетково-пакувальної продукції, розвитку нових методів захисту і контролю захищеної поліграфічної продукції.

Мета та задачі дослідження

Широке поширення контрафактних товарів викликає необхідність в створенні методів боротьби з фальсифікацією. Передусім, природно, це впровадження нових і досконаліших захисних елементів і методів їх контролю [12, 15, 16]. Проте не меншу значущість має інформування споживача про певні види захисту товару і посилення відповідальності за фальсифікацію.

Враховуючи, що фальсифікація етикетково-пакувальної продукції є непрямою, слід зазначити, що фальсифікаторам загрожує покарання вже за первинну підробку. Міра відповідальності визначається від обсягів виготовлених та розповсюджених фальсифікатів, а також від шкоди, нанесеної цією продукцією споживачам і виробникам оригінальної продукції. Говорячи виключно про етикетково-пакувальну продукцію, в нашій країні ухвалені законопроекти, що вимагають обов'язкових захисних елементів лише на упаковці лікарських препаратів і наявності акцизних марок на відповідній продукції. Тобто, зараз виробник повинен сам піклуватися про захист власної продукції, отже, потрібні розробка і прийняття актів і законів, регулюючих сферу етикетково-пакувальної промисловості в повнішому обсязі.

Захисні елементи і методи їх контролю мають бути застосовані на усіх етапах виробництва етикеток та упаковок. На додрукарській стадії при створенні оригінал-макету впроваджуються деякі елементи захисту продукції [14].

Не менш важливо перевіряти правильність виготовлення макету і застосування усіх елементів [13].

Дизайнер має бути упевнений, що він використовує правильний логотип, мікротекст, приховане зображення, колірне поєднання і інші способи захисту. Таким чином, актуальне створення засобу контролю – використання цих елементів [12, 15].

Основна частина

1 Аналіз методичної складової дослідження

Ситуація, що склалася на ринку поліграфічної продукції, показує, що такі поняття як конкуренція і фальсифікація зустрічаються постійно. Практично будь-яка система захисту може бути підроблена, проте високий рівень захисту вимагає великих витрат від виробника, що не завжди можливо і доцільно. Іноді продукція зобов'язана мати захисні елементи, наприклад, упаковка медикаментів. Виробник кінцевого продукту не завжди належним чином обізнаний про принципи захисту упаковки, отже, їх повинні знати, уміти відрізнити і застосувати виготівники оригінал-макетів етикетки і упаковки. Проте не усі способи захисту є зручними для споживача, і навіть фахівцеві іноді потрібне певне устаткування для перевірки наданого зразка.

З цього виходить необхідність оперативного контролю продукції, наявність усіх необхідних елементів системи захисту упаковки і етикетки, а також перешкода випуску фальсифікованих зразків. Для цього треба виявити причини появи підробних зразків.

Причинами можуть бути цілеспрямоване виробництво фальсифікату і виробництво за допомогою посередників. Найчастіше шахраї не мають спеціального устаткування для повного циклу виробництва продукції, а діють через посередників. Проміжна організація, наприклад виробник кліше, може бути не обізнана про достовірність об'єкту друку, і, хоча це не позбавляє його від відповідальності, фальсифікат, проте, виробляється. А також посередник не завжди має можливість перевірити надані відомості на предмет фальсифікації. У цьому полягає проблема, яку необхідно вирішити.

Метою дослідження є оцінка способів і методів захисту етикетково-пакувальної продукції і створення програмного засобу їх контролю на додрукарській стадії виробництва, оскільки саме на цій стадії це найбільш затребувано і практично. В практичній частині дослідження розглянуто програмний засіб, дія якого заснована на розробленому алгоритмі, і який дозволить виявити фальсифікований зразок у порівнянні його з оригіналом.

1.1 Теоретичні засади алгоритму задачі

Концепція створюваного скрипта ґрунтується на порівнянні зображень, одне з яких свідомо є оригінальним. Відомо, що зображення бувають векторними і растровими. Проте отримати для порівняння векторний зразок можливо не завжди. Растрове зображення є масивом пікселів. Кожен елемент цього масиву зберігає необхідну інформацію щодо кольору. Кількість колірних координат, що

зберігаються в пікселі, залежить від колірного простору растрового зображення. Оскільки дані зображення є поліграфічними оригінал-макетами, то розробку скрипта проведено для колірної моделі CMYK.

Порівняння зображень для контролю наявності захисних елементів має за мету пошук відмінностей на двох зображеннях. Для цього кожен піксель оригіналу необхідно порівняти з відповідним йому пікселем наданого для перевірки зображення. Якщо усі пікселі попарно співпадають, то зображення ідентичні, і можна стверджувати, що надане для перевірки зображення не є фальсифікатом, а також має усе необхідні захисні елементи.

Елементарною перевіркою збігу є віднімання значень пікселів. Проте слід зазначити, що кожен піксель зберігає як мінімум чотири числові значення – колірні координати Cyan, Magenta, Yellow, Black. Таким чином, при відніманні двох пікселів необхідно порівнювати чотири пари значень. При абсолютному збігу результатом віднімання є нуль, що означає рівність колірних значень пікселів. Нульове значення усіх чотирьох координат CMYK відповідає білому кольору, що робить можливим зручне візуальне спостереження результатів віднімання. При цьому значення неспівпадаючих пікселів може варіюватися.

Особливістю людського зору є сприйняття картини в цілому, а також адаптація до білого кольору: якщо найсвітліший відтінок зображення близький до білого, то він може сприйматися як білий. Отже, окрім візуального відображення результатів віднімання, потрібний аналіз отриманого зображення і виведення числового значення, що дозволяє оцінити міру збігу порівнюваних зображень. Програмний комплекс MatLab надає можливості як аналітичної, так і візуальної оцінки зображень.

Оскільки кожен піксель результуючого зображення містить інформацію про колірні координати, які є підсумком попарного віднімання чотирьох колірних значень, то надається можливість не лише абсолютного, але і поканального порівняння зображень. Це робить реальним аналіз колірних характеристик і відхилень даних зображень, що важливо для елементів захисту.

Для дослідження і оцінки застосування захисних елементів в алгоритмі має бути передбачене завдання міри чутливості, під якою слід розуміти поріг ідентичності зображень. Тобто надається можливість приймати зображення за ідентичні, якщо їх різниця нижча за порогове значення. Налаштування порогу чутливості до відмінностей повинне вироблятися вручну, і залежить від конкретних ситуацій, проте, слід зазначити, що абсолютну різницю зображень дає виключно використання нульового порогового значення.

Під час написання алгоритму також було враховано, що аналізуватися і порівнюватися може велика кількість зображень. Порівняння і аналіз зображень вручну – це досить трудомісткий процес, який вимагає значних витрат часу. Це означає, що потрібно додати автоматичну пакетну обробку зображень.

На підставі представлених вимог і цілей складений алгоритм програмного засобу порівняння зображень:

– визначення адреси локального розташування оригінальних і потенційно фальсифікованих зображень;

– визначення відповідності зображень для перевірки, необхідним оригіналам;

– безпосереднє віднімання відповідних зображень;

– розрахунок абсолютної і поканальної різниці зображень на підставі даних результуючого зображення;

– збереження результуючого зображення, ім’я якого містить дані математичного аналізу ідентичності порівнюваних зображень.

1.2 Графічне представлення алгоритму

Відповідно до представленого алгоритму для наступного написання програмного коду було створено блок-схему процесу.

Блок-схема – поширений тип схем (графічних моделей), що описують алгоритми або процеси, в яких окремі кроки зображені у вигляді блоків різної форми, сполучених між собою лініями, що вказують напрям послідовності [10].

Оскільки MatLab є інтерпретованою мовою програмування, ця графічна модель наочно представляє дію алгоритму і взаємозв’язок процесів усередині нього. Блок-схема складається згідно ГОСТ 19 701-90 [10]. В даному випадку для виконання алгоритму досить використання таких складових елементів блок-схеми, як процеси, зумовлені процеси, цикли, рішення і лінійні символи.

Для роботи з конкретною парою зображень з масиву наданих зображень алгоритм програми припускає створення декількох вкладених циклів, що визначають відповідність оригінальних та наданих для перевірки зображень та подальші обчислення міри збігу вибраної пари.

Представлена на рис. 1 блок-схема відображує послідовність виконання дій та особливості безпосереднього відображення цих дій в програмному коді. Блок-схема дозволяє продемонструвати логічну структуру програмного коду та уникнути використання надмірних функцій і процедур.

Процес визначення абсолютного і поканального збігу порівнюваних зображень також може бути відбитий в блок-схемі. Для зручності читання на схемі відображений не лише сам процес, але і межі циклу, якому процес належить. Схема приведена на рис. 2.

Процес, відображений на останній блок-схемі, є виконанням дій для визначення абсолютного і поканального збігу конкретних відповідних порівнюваних зображень. Відповідно до розробленого алгоритму, цей процес виконується після віднімання зображень, і дії процесу виконуються вже над зображенням, отриманим в результаті віднімання.

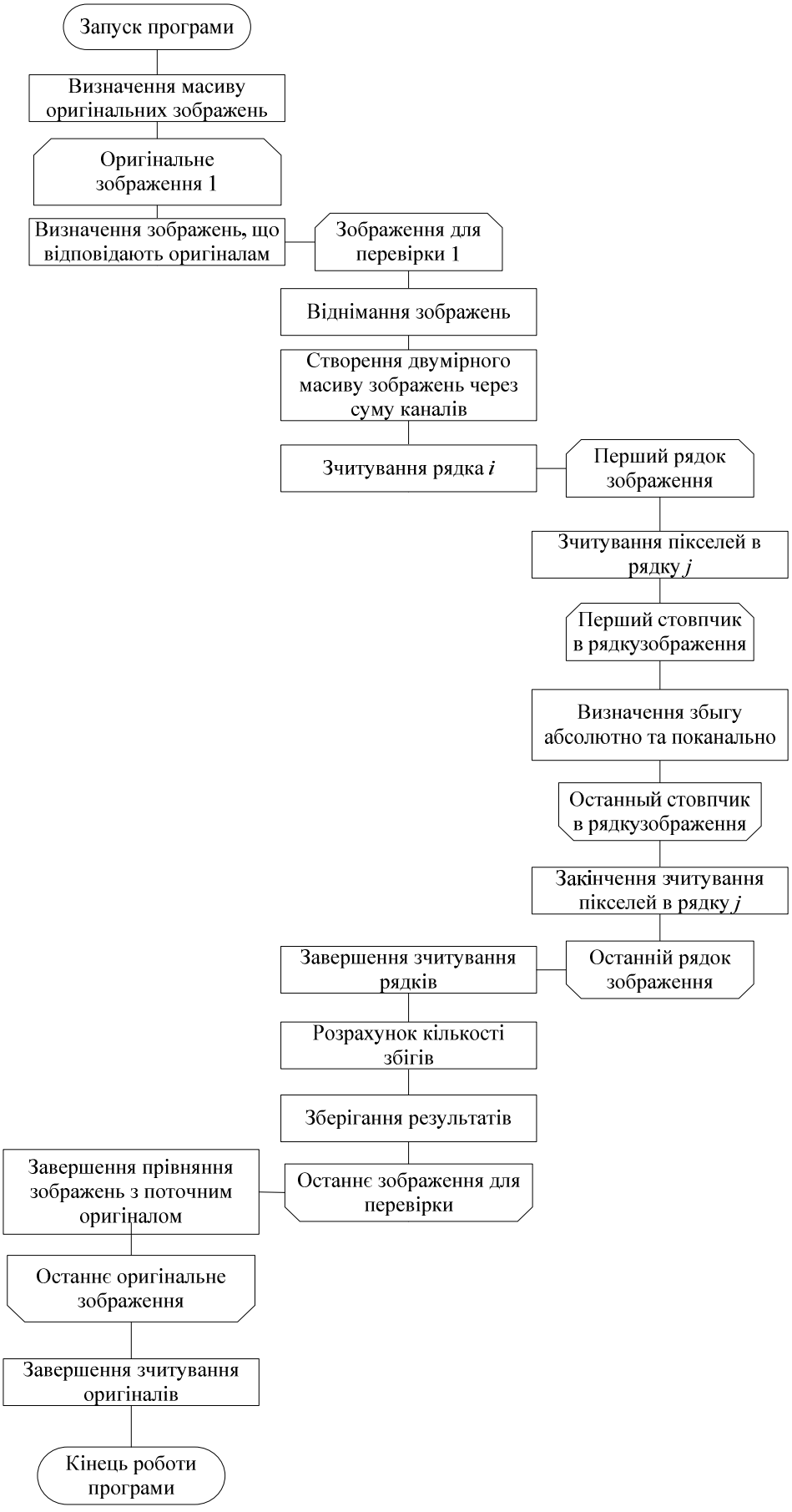


Рисунок 1 – Загальна блок-схема процесу

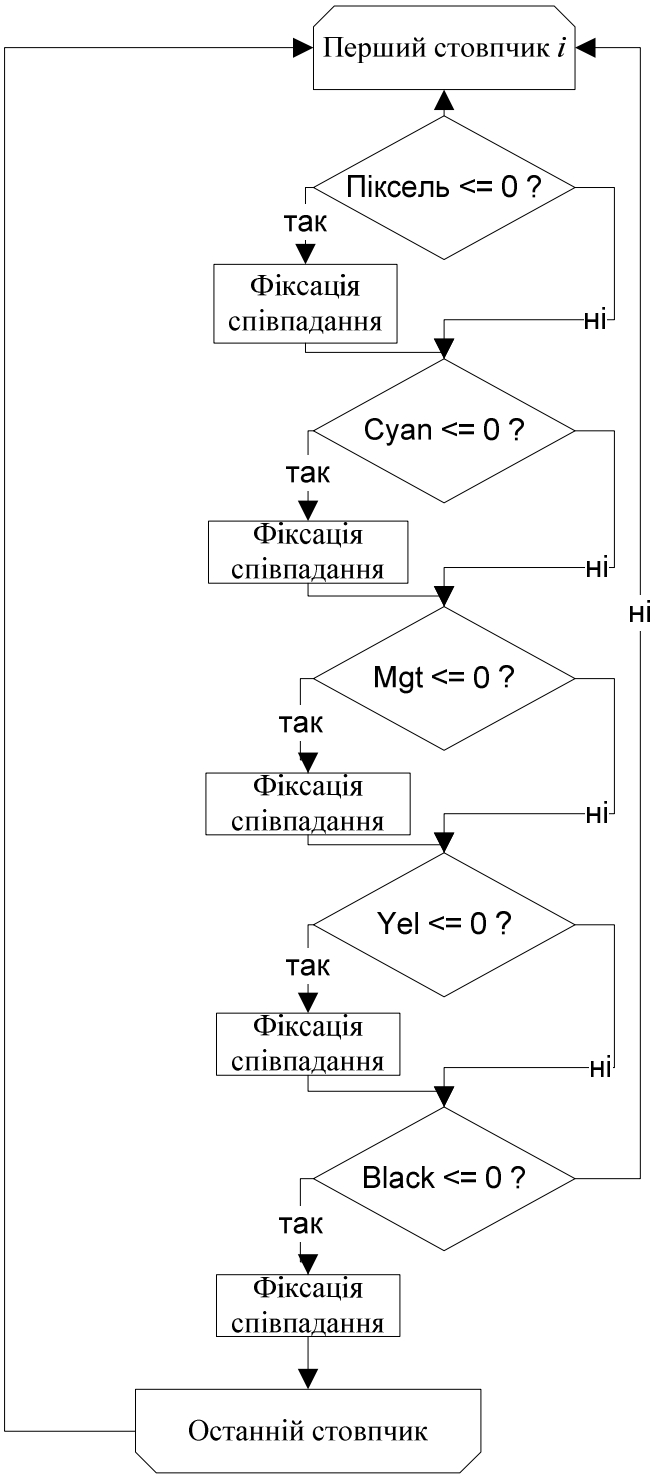


Рисунок 2 – Блок-схема частини процесу щодо визначення збігів

Визначення поточного пікселя виконується циклом, в який входить цей процес. Цикл послідовно працює з пікселями поточного рядка результуючого зображення. Збіг відповідних пікселів оригіналу і зображення для перевірки, в результуючому зображенні дає білий колір. Таким чином, якщо колірні координати пікселя результуючого зображення дорівнюють (0, 0, 0, 0), то вважається, що пікселі співпали. Результуюче зображення є матрицею розміру MxNx4, що відбиває наявність чотирьох колірних каналів.

Отже, для перевірки абсолютного збігу потрібне звернення до одного і того ж пікселя чотири рази. Проте слід зазначити, що сума нулів також дорівнює нулю. Це означає, що перетворивши масив MxNx4 в двовимірний масив можна звертатися до кожного пікселя один раз. Для цього до коду додано операцію лінійного складання матриць, які становлять результуюче зображення. Результатом складання є масив chSumArray.

Процес визначення абсолютного збігу пікселів складається з рішення, результат якого закладений умовою. Ця умова порівнює значення конкретного пікселя перетвореного масиву chSumArray результуючого зображення, що належить рядку *i* та стовпцю *j* масиву, з пороговим значенням, яке, в даному випадку, дорівнює нулю. Для завдання порогу не повинна використовуватися спеціальна змінна, оскільки враховується можливість порівняння при різних порогах для наступного порівняння окремих каналів. Якщо значення пікселя менше або дорівнює значенню порогу, збіг фіксується. При невиконанні вказаної умови, фіксація не відбувається, і програма переходить до наступного етапу виконання. Таким чином, в циклі фіксуються збіги пікселів по всьому зображенню.

Поканальне порівняння відбувається за інших умов. В цьому випадку не можна працювати з перетвореним двовимірним масивом, оскільки при цьому втрачається інформація про значення окремих колірних складових. Таким чином, умовою збігу пікселів є: значення каналу пікселя результуючого зображення currentResult менше або рівне 0, де currentResult є результатом віднімання поточних зображень, що звіряються, Для визначення каналу потрібне звернення currentResult (*i*, *j*, 1) що означає виклик матриці *i*xw, що відображує значення першого каналу (cyan) результуючого зображення. Порівняння з порогом чутливості відбувається аналогічно подібному порівнянню при визначенні абсолютного збігу. При виборі другого, третього і четвертого каналу викликаються канали magenta, yellow і black відповідно. Використовувані умови порівняння аналогічні умові блакитного каналу і відрізняються лише координатою звернень, визначальної номер матриці, що викликається, в масиві MxNx4.

Цикл завершується після повної обробки поточного зображення. Оскільки цикл є вкладеним, то він повторюється для усіх порівнюваних зображень.

Представлені графічні моделі алгоритму у вигляді блок-схем дозволяють створити програмний код і простежити коректність його виконання.

1.3 Аналіз програмного коду

Відповідно до представленого алгоритму і блок-схем, що демонструють його роботу, створено програмний код. Програмний пакет MatLab надає усі необхідні інструменти і можливості для написання і відладки коду. При складанні алгоритму врахована вірогідність різної кількості зображень оригінальних і таких для перевірки.

Таким чином, скрипт надає можливість роботи з такими типами відношення кількості оригіналів і зображень для перевірки:

– один до одного (порівнюється два зразки, один – оригінальний, другий – наданий для перевірки);

– один до багатьох (одному оригіналу відповідає певна кількість зображень для перевірки);

– багато до багатьох (кожен з кількох наданих оригіналів має різну незалежну кількість відповідних зображень для перевірки).

Засоби MatLab дозволяють порівняння зображень з колірним простором CMYK у форматі tif, який є широко використовуваним в поліграфічній галузі.

Основним завданням в програмній реалізації алгоритму є коректне виконання віднімання заданих зображень. Пакет розширення MatLab Image Processing Toolbox надає достатню кількість готових функцій, які виконують віднімання зображень. Проте необхідно проаналізувати, які вони мають відмінності, і яка саме функція відповідає завданню дослідження. Серед різних функцій обробки пари зображень в Image Processing Toolbox існують такі функції, як imshowpair, imdivide, imsubtract, imabsdiff [11]. Дія усіх розглянутих функцій перевірена на практиці і відповідає теоретичному опису. При оцінці дії функції проводилася візуальна оцінка результатів, ручний перерахунок значень за допомогою програмного пакету Adobe Photoshop Extended CS5. а також перевірка дії функції на відповідність створеному алгоритму.

Поверненим результатом функції imshowpair є візуалізація відмінностей двох зображень. Проте це можливо виключно у тому випадку, коли зображення є матрицею розміром MxN або MxNx3, що робить неможливим порівняння зображень з колірною моделлю CMYK без попереднього перетворення матриць каналів в бінарні зображення або перетворення самої колірної моделі. В переході з одного колірного простору в інше відбувається часткова втрата кольорів, оскільки колірне охоплення різних моделей відрізняється Якщо застосовувати це перетворення, стає неможливим поканальне порівняння зображень і аналіз колірних відхилень.

Функція imdivide ділить кожен піксель початкового зображення на відповідний йому піксель порівнюваного зображення. Під час виконання команди створюється масив, що містить результати ділення. Це дозволяє розрахувати процентне співвідношення збігу оригінального і потенційно фальсифікованого зображення, проте не дає можливості коректної візуалізації отриманого масиву. Так ділення двох ідентичних пікселів дасть результат у вигляді 1 cyan, 1 magenta, 1 yellow 1 black.

Imsubtract є поверненням двовимірного масиву, яке виходить в результаті віднімання одного зображення з іншого. Особливістю imsubtract є залежність від порядку порівняння, тобто при порівнянні оригіналу із порівнюваним зображенням і навпаки результат буде різним. Це не відповідає складеному алгоритму, оскільки результат візуального відображення результатів буде невірним.

Результатом дії функції imabsdiff є масив значень абсолютної різниці попарно відповідних один одному пікселів порівнюваних зображень. Під абсолютною різницею в даному випадку є результат віднімання відповідних значень за модулем. Ця функція підтримується в колірному просторі CMYK, дозволяє коректно здійснювати необхідні обчислення і коректно відображувати результуючі зображення.

Таким чином, функція imabsdiff оптимально підходить для використання в створюваному програмному засобі і повністю відповідає закладеній в алгоритмі концепції. Таким чином немає необхідності створювати власну функцію порівняння зображення за допомогою віднімання, що також підтверджує правильність вибору MatLab.

За коректне виконання етапів алгоритму відповідають створені в програмному коді змінні, функції і процедури. В ході написання скрипта враховувалися такі параметри як відповідність алгоритму поставленій задачі, читабельність коду, відсутність надмірних дій, ефективність, швидкодія і отримання очікуваних результатів. Для визначення цих параметрів слід детально розглянути код скрипта.

Блок визначення локального розташування порівнюваних зображень і шляху збереження результату порівняння виконується такими функціями:

```
originFolder='D:\matlabImages\origin';
copiesFolder='D:\matlabImages\copies';
resultFolder='D:\matlabImages\result'.
```

Кожному зображенню відводиться певне місце зберігання. До змінних, які задають адресу розташування файлів, програма звертається для створення пар порівнюваних зображень.

Функція originImagesArray = dir(fullfile(originFolder', *.tif')) використовує вже створену змінну originFolder і відповідає за створення вибірки оригінальних зображень формату tif, розташованих за заданою адресою.

Визначення кількості оригінальних зображень originArray Length = length(originImagesArray) потрібне для створення масиву оригіналів, відповідно до яких буде створений масив зображень для перевірки.

Змінна rowNumber відповідає за наступне збереження отриманої інформації у файлі Excel.

Таким чином, створюється цикл з межами for i=1 : originArrayLength, що визначає об'єкт подальших дій (масив оригінальних зображень). Всі наступні процеси протікають в цьому циклі і завершуються з закінченням масиву оригіналів.

Дія originsPath = strcat (originFolder, '\', originImagesArray(i). name) об'єднує локальне розташування і ім'я файлу, таким чином, отримуючи повний шлях до файлу. Далі відбувається зчитування конкретного оригіналу currentOriginI = imread(originsPath). Для визначення зображень для перевірки, що відповідають поточному оригіналу, виконується привласнення

```
[~, originName, ext] = fileparts(originImagesArray(i).name)
```

тобто відокремлення імені файлу оригіналу від розширення та створення масиву зображень для перевірки порівнянням саме з цим оригіналом

```
copiesImagesSubArray = dir (fullfile (copiesFolder, strcat (originName'*')))).
```

При цьому ім'я зображення для перевірки має починатися з імені оригіналу.

Для масиву зображень для перевірки створюється цикл з межами for j=1: length(copiesImagesSubArray), що визначає конкретне зображення у нинішній момент часу для перевірки. Процес визначення виконується таким чином:

```
copiesPath = strcat(copiesFolder',\', copiesImagesSubArray(j).name);
currentCopyI=imread(copiesPath);
```

Оскільки в попередніх процесах визначена конкретна пара порівнюваних зображень, то можна безпосередньо виробляти віднімання:

```
currentResult = imabsdiff(currentOriginI, currentCopyI).
```

Відповідно до алгоритму виконується перетворення результуючого зображення в двовимірний масив, шляхом лінійного збільшення матриць, що зберігають значення каналів зображення.

```
chSumArray = currentResult(:,:, 1) + currentResult(:,:, 2) + currentResult(:,:, 3) + currentResult(:,:, 4).
```

Виконання процесу визначення абсолютного і поканального збігу вимагає ініціалізації нових змінних:

– whiteAmount=0; – змінна, що зберігає дані про абсолютний збіг;

– cAmount=0, mAmount=0, yAmount=0, kAmount=0 – змінні, що зберігають кількість збігів по каналах cyan, magenta, yellow, black відповідно.

Функція pixelAmount = size (chSumArray, 1) * size (chSumArray, 2) визначає кількість пікселів результуючого зображення для наступних розрахунків.

Оскільки виробляється повний аналіз зображення потрібно розрахунок для кожного пікселя. Для цього створюються 2 підцикли з межами for *i*=1: size(chSumArray, 1) – визначають рядок масиву зображення і for *j*=1: size(chSumArray, 2) – що визначають стовпець масиву. Таким чином, кожен піксель має унікальні координати *i* і *j*.

У рамках цих циклів виконуються такі дії:

```
if (chSumArray(i, j)==0)
    whiteAmount=whiteAmount+1;
end
if(currentResult(i, j, 1)<=0)
  cAmount=cAmount+1;
end
if(currentResult(i, j, 2)<=0)
  mAmount=mAmount+1;
end
if(currentResult(i, j, 3)<=0)
```

```
    yAmount=yAmount+1;
end
if(currentResult(i, j, 4)<=0);
  kAmount=kAmount+1.
end.
```

Абсолютні (збіги по усіх чотирьох каналах) і поканальні значення пікселя порівнюються зі встановленим пороговим значенням. Для досягнення точної ідентифікації ідентичності відповідних пікселів порівнюваних зображень, порогове значення має дорівнювати нулю. Проте можливе ручне коригування порогу чутливості. Після завершення циклу поточне зображення повністю проаналізоване на предмет абсолютних і поканальних збігів, проте ці збіги виражені лише в кількісному вигляді. Ця інформація недостатня як для сприйняття, так й для подальшого аналізу.

Обчислення абсолютного збігу виконується за пропорцією whitePercentage = (whiteAmount/pixelAmount)*100. Таким чином, результат збігу по усій площі порівнюваних зображень виражається у відсотках. Аналогічний розрахунок виробляється для усіх колірних каналів СМУК, результати привласнюються змінним С, М, У, К.

Збереження результатів має передбачати чітку відповідність із зображенням для перевірки. Для цього застосовується процедура, аналогічна видокремленню імені оригіналу.

```
[~,copyName, copyExt]=fileparts(copiesImagesSubArray(j).name).
```

Значення, що обчислюються в MatLab, мають числовий тип даних, що не дозволяє записувати їх в будь-який файл без обробки. Для фіксації цих значень необхідно перетворити тип даних в рядковий. Дія виконується за допомогою функції num2str, яка по черзі застосовується до змінних, що містять процентне вираження збігу зображень.

Збереження результатів віднімання зображень і наступних обчислень виконує функція imwrite(currentResult, strcat(resultFolder’, \’, copyName, ‘_’, str, ‘_C ‘, cyan’, _M ‘, magenta’,_Y ‘, yellow’,_K ‘, black, copyExt)).

Для зручної роботи з информацією, яка отримана в результаті обробки зображень, усі дані зберігаються у файлі Excel, що є таблицею даних.

```
excel=‘D:\matlabImages\result\MyBook.xls’
x=[whitePercentage]; % абсолютний збіг
xc=[C];
xm=[M];
xy=[Y];
xk=[K].
```

Дані змінні визначають локальне розташування файлу – D:\matlabImages\result\MyBook.xls і значення для наступного запису.

```
if (x==100);
      xres=[‘+’];
end
```

```
if (x~=100);
      xres=['-'];
end
```

В результаті представлених рішень, програма автоматично визначає зображення як ідентичне оригіналу або ні, і заносить відповідний висновок в таблицю знаком + або - відповідно.

Далі виконується запис конкретних значень в таблицю, включаючи імена рядків і стовпців, що формують таблицю.

```
variables={'ABS ',' C ','M',' Y ',' K ',' result'}; % заголовки стовпців
xlswrite(excel, variables', b1: g1'); % записуємо заголовки стовпців
points={copyName}; % заголовки рядків
xlswrite(excel, points,['a' num2str(rowNumber) ': a' num2str(rowNumber)]); % записуємо заголовки рядків
xlswrite(excel, x,['b' num2str(rowNumber) ': b' num2str(rowNumber)]); % запис абсолютного збігу
xlswrite(excel, xc,['c' num2str(rowNumber) ': c' num2str(rowNumber)]);
xlswrite(excel, xm,['d' num2str(rowNumber) ': d' num2str(rowNumber)]);
xlswrite(excel, xy,['e' num2str(rowNumber) ': e' num2str(rowNumber)]);
xlswrite(excel, xk,['f' num2str(rowNumber) ': f' num2str(rowNumber)]);
xlswrite(excel, xres,['g' num2str(rowNumber) ': g' num2str(rowNumber)]);
rowNumber=rowNumber+1;
```

Функція rowNumber=rowNumber+1 відповідає за запис даних поточного зображення у вільному рядку, що йде за попередньою заповненою.

Після збереження результату для поточної пари порівнюваних зображень, за умовами циклу for j=1 : length(copiesImagesSubArray) програмний код виконується для наступної пари, в якій оригінал залишається тим самим, а зображення для перевірки, замінюється на наступне. Якщо усі зображення для перевірки, відповідні поточному оригіналу вже пройшли усе перетворення, то за умовами циклу for i=1 : originArrayLength, код виконується для наступного оригінального зображення, до тих пір, поки не будуть використані все наявні оригінали.

Таким чином створений програмний код відповідає усім заявленим вимогам, не має надмірностей і при компіляції видає необхідний результат. Для коректного виконання програмного коду необхідно виконання таких вимог:

– зображення мають бути збережені у форматі tif з використанням колірного простору СМУК;

– порівнювані зображення розсортовані по заздалегідь створених теках, відповідних вказаних в програмному коді;

– ім'я зображення для перевірки повинне починатися з імені відповідного оригіналу, якщо зображення для перевірки одне – імена можуть співпадати.

Перевагою створеної програми є автоматична обробка, висока швидкодія, можливість роботи з великою кількістю файлів. Алгоритм, використовуваний в роботі, також підходить для порівняння зображень в колірному просторі RGB, що розширює можливості порівняння графічних файлів, що мають формат,

відмінних від tif. Проте в даній ситуації актуальне використання зображень з колірною моделлю CMYK.

2 Тестування роботи програми в різних ситуаціях

Для об'єктивної оцінки діяльності програми потрібно попереднє тестування. Завдяки тестуванню можна з'ясувати можливості використання програми, а також умов, в яких її виконання буде коректним і найбільш ефективним.

Уточнення вимог, необхідних для коректного виконання програми вимагає тестування програмного коду в різних ситуаціях, виникнення яких можливе при експлуатації програми. При тестуванні розглянуті наступні ситуації:

– зображення представлені в цифровому вигляді, мають однаковий розмір і роздільну здатність;

– зображення представлені в цифровому вигляді, мають різний розмір і однакову роздільну здатність;

– зображення представлені в цифровому вигляді, мають однаковий розмір і різну роздільну здатність;

– зображення представлені в аналоговому вигляді і вимагають оцифрування.

Зображення, що мають однакову роздільну здатність і розмір, при цьому представлені в цифровому виді, повністю відповідають складеному алгоритму, тому результат їх порівняння очікуваний та коректний.

Виконання порівняння зображень, що мають різні розміри або роздільну здатність, за допомогою цього програмного засобу неможливо, оскільки використовується порівняння відповідних матриць. Програмно для зображень можна задати ці параметри однаковими, MatLab Image Processing Toolbox надає для цього усі необхідні функції. Проте зміна розміру зображень, так само як і зміна дозволу, спричиняє за собою інтерполяцію значень сусідніх пікселів, що викликає спотворення зображення.

Зміна роздільної здатності приводить до інтерполяції значень сусідніх пікселів, що викликає спотворення зображення, Особливо ці спотворення, відбиваються на елементах захисту, особливо на елементах, що мають малий розмір. Отже, і інформація, отримана від порівняння, буде недостовірною. Слід зазначити, що якщо роздільна здатність і розмір зображень, що зберігаються в цифровому виді не співпадають, то зображення мають відмінності.

На стадії контролю захисних елементів це особливо важливо, оскільки відмінність цих параметрів свідчить про помилковість виконання оригінал-макету. Застосувавши попередню обробку за допомогою Adobe Photoshop або інших програмних пакетів, можливо привести зображення до формату, придатного для порівняння. Ці дії можна автоматизувати за допомогою action Photoshop. Для цього виконується наступна послідовність дій:

– створення в палітрі Actions нову операцію (new action), завдання необхідних параметрів: ім'я, поєднання клавіш, колір (якщо потрібно);

– запуск запису створеного Action;

– Image – Image Size – Resolution. За допомогою цієї команди задається роздільна здатність, що відповідає роздільній здатності оригіналу. Таким чином, компенсуються можливі відмінності роздільної здатності, що встановлюються пристроями введення інформації. Підключаються функції: зберегти стилі, зберегти пропорції;

– File – Automate – Fit image. Дія встановлює розмір зображення, згідно з параметрами еталонного оригінального зображення, які відомі заздалегідь;

– завершення запису Action. Action може застосовуватися як для перетворення оригінальних зображень, так і зображень для перевірки. Мінімальне встановлюване дозволі повинно дорівнювати 300 dpi, оскільки нижче значення буде недостатньо для відображення елементів захисту.

Проте оскільки інформація спотворюється, зроблений висновок про перевагу отримання зображень за допомогою однакових методів, аби в результаті вони мали однакову роздільну здатність і розмір.

Зображення, представлені у вигляді поліграфічних зразків, потребують оцифрування за допомогою сканера. Не зважаючи на те, що зараз досягнутий порівняно високий рівень техніки, він недостатній для подібних порівнянь. Частково це обумовлено застосуванням методів антисканерного захисту. Таким чином можна зробити висновок лише про те, що якщо відсканований зразок повністю відповідає поліграфічному, проте заздалегідь відомо, що в оригінал-макеті застосовані антисканерні елементи захисту – можна стверджувати, що зразок для перевірки буде визначений як підробка. Однак основним недоліком сканерів є перетворення колірних моделей. Отримання достовірних даних про колір при цьому неможливе, оцифрування може бути застосовано лише у разі однакових умов сканування оригінала і зразка для перевірки: використання одного сканера із заданими однаковими параметрами, використання однакових профілів. Слід враховувати також можливі спотворення в ході сканування, наприклад зсув або поворот. Таким чином використання зображень, отриманих шляхом сканування небажано, оскільки наявний високий рівень погрішності, що під час порівняння елементів захисту може стати критичним. Компенсація за допомогою порогу чутливості в даному випадку також недостатньо ефективна.

2.1 Опис взаємодії з програмою

Коректне використання програми можливе у тому випадку, якщо усі необхідні зображення підготовлені і розташовані у відповідних теках. Рисунок 3 демонструє створені теки origin, copies, result відповідно для оригінальних зображень, зображень для перевірки та результатів.

Тека origin містить оригінальні зображення, що приймаються за еталонні при порівнянні. Кожен файл в теці має унікальне ім’я. Приклад розміщення оригіналу в теці і відкритого оригіналу представлений на рисунку 4.

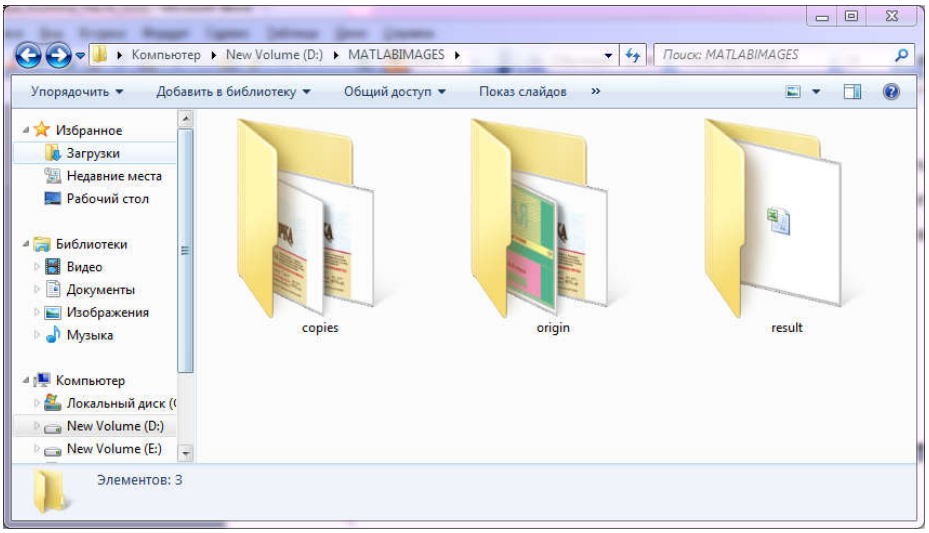


Рисунок 3 – Теки розташування файлів

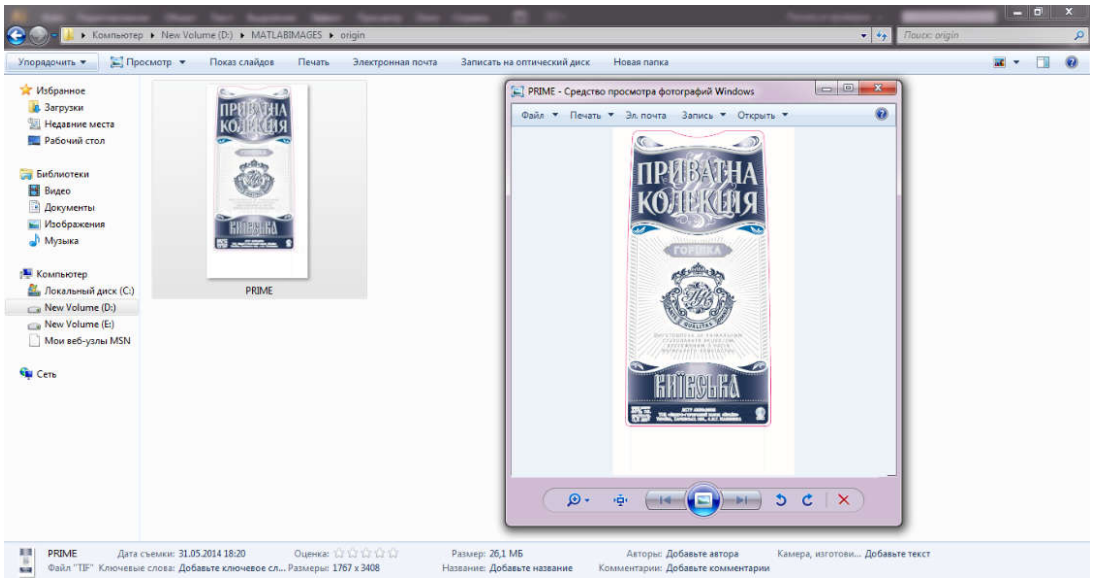


Рисунок 4 – Вміст теки origin

Зображення для перевірки, відповідні оригінальним, знаходяться в теці copies. Відповідність зображень визначається за ім’ям. Початок імен оригіналу і зображень для перевірки мають співпадати. Таким чином здійснюється можливість порівняння декількох зображеньдля перевірки з одним оригіналом. На рисунку 5 зображена тека copies і файли, що містяться в ній.

Наступним кроком є відкриття файлу MatLab, що містить необхідний програмний код (рис. 6).

Якщо зображення підготовлені, розташовані у відповідних теках і користувача влаштовує заданий поріг чутливості, то досить запустити код на виконання. Це можна зробити двома способами:

– запуск за допомогою кнопки Run включає виконання коду;

– запуск за допомогою кнопки Run and Time не лише включає виконання коду, але і запускає profiler, що виводить значення швидкості поетапного виконання коду в таблиці.

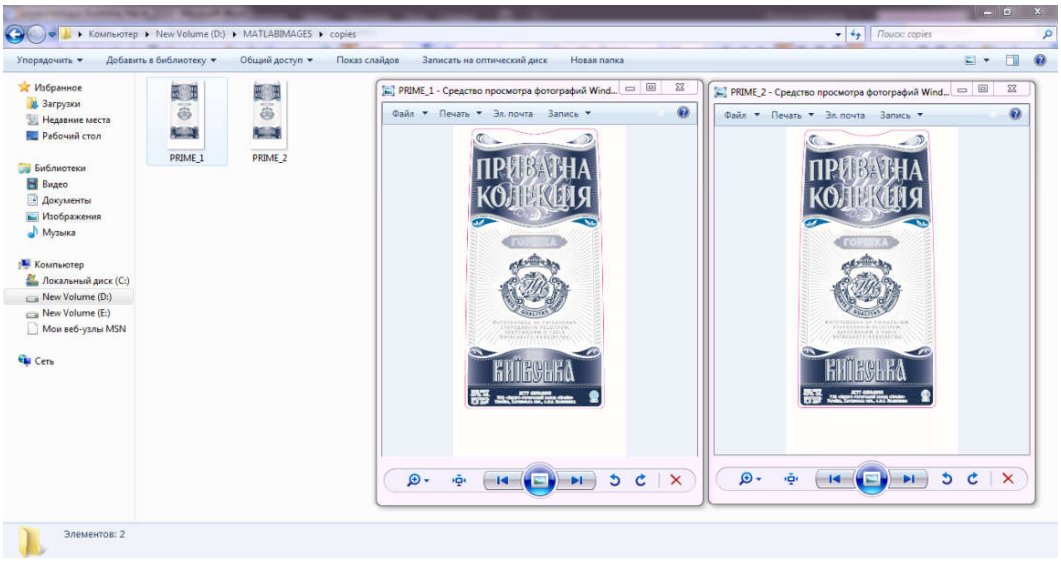
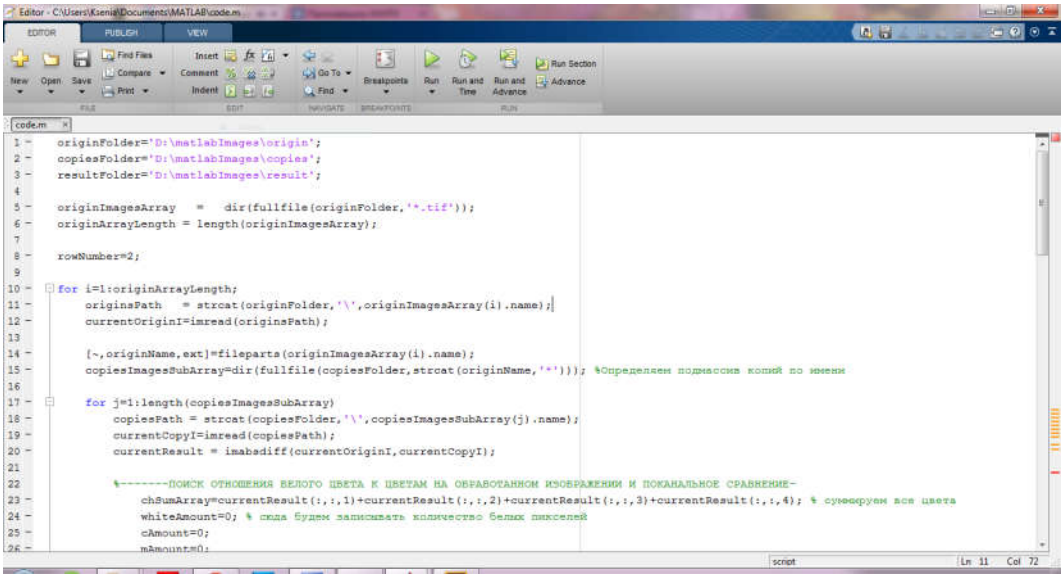


Рисунок 5 – Вміст теки copies



```
originFolder='D:\matlabImages\origin';
copiesFolder='D:\matlabImages\copies';
resultFolder='D:\matlabImages\result';

originImagesArray   =   dir(fullfile(originFolder,'*.tif'));
originArrayLength = length(originImagesArray);

rowNumber=2;

for i=1:originArrayLength;
    originPath   = strcat(originFolder,'\',originImagesArray(i).name);
    currentOriginI=imread(originPath);

    [~,originName,ext]=fileparts(originImagesArray(i).name);
    copiesImagesSubArray=dir(fullfile(copiesFolder,strcat(originName,'*'))); %Определяем подмассив копий по имени

    for j=1:length(copiesImagesSubArray)
        copiesPath = strcat(copiesFolder,'\',copiesImagesSubArray(j).name);
        currentCopyI=imread(copiesPath);
        currentResult = imabsdiff(currentOriginI,currentCopyI);

        %-------ПОИСК ОТНОШЕНИЯ БЕЛОГО ЦВЕТА К ЦВЕТАМ НА ОБРАБОТАННОМ ИЗОБРАЖЕНИИ И ПОКАНАЛЬНОЕ СРАВНЕНИЕ-
            chSumArray=currentResult(:,:,1)+currentResult(:,:,2)+currentResult(:,:,3)+currentResult(:,:,4); % суммируем все цвета
            whiteAmount=0; % сюда будем записывать количество белых пикселей
            cAmount=0;
            mAmount=0;
```

Рисунок 6 – Програмний код в MatLab

Якщо теки, що містять необхідні зображення, знаходяться у іншому місці, необхідно або перенести їх, або відкоригувати наступні строчки коду :

originFolder=‘D:\matlabImages\origin’;
copiesFolder=‘D:\matlabImages\copies’;
resultFolder=‘D:\matlabImages\result’;
excel=‘D:\matlabImages\result\MyBook.xls’.

Якщо заданий поріг чутливості не влаштовує користувача, також потрібно ручне коригування значення в рядках:

if(chSumArray(*i*, w)<=0)
if(currentResult(*i*, w, 1)<=0)
if(currentResult(*i*, w, 2)<=0)
if(currentResult(*i*, w, 3)<=0)
if(currentResult(*i*, w, 4)<=0)

У результаті при натисненні кнопки Run або Run and Time, програмний код виконується і результуючі зображення записуються в теку result, яка повинна вже містити файл Excel з ім'ям MyBook. Приклад вмісту теки result після виконання коду наведений на рисунку 7.

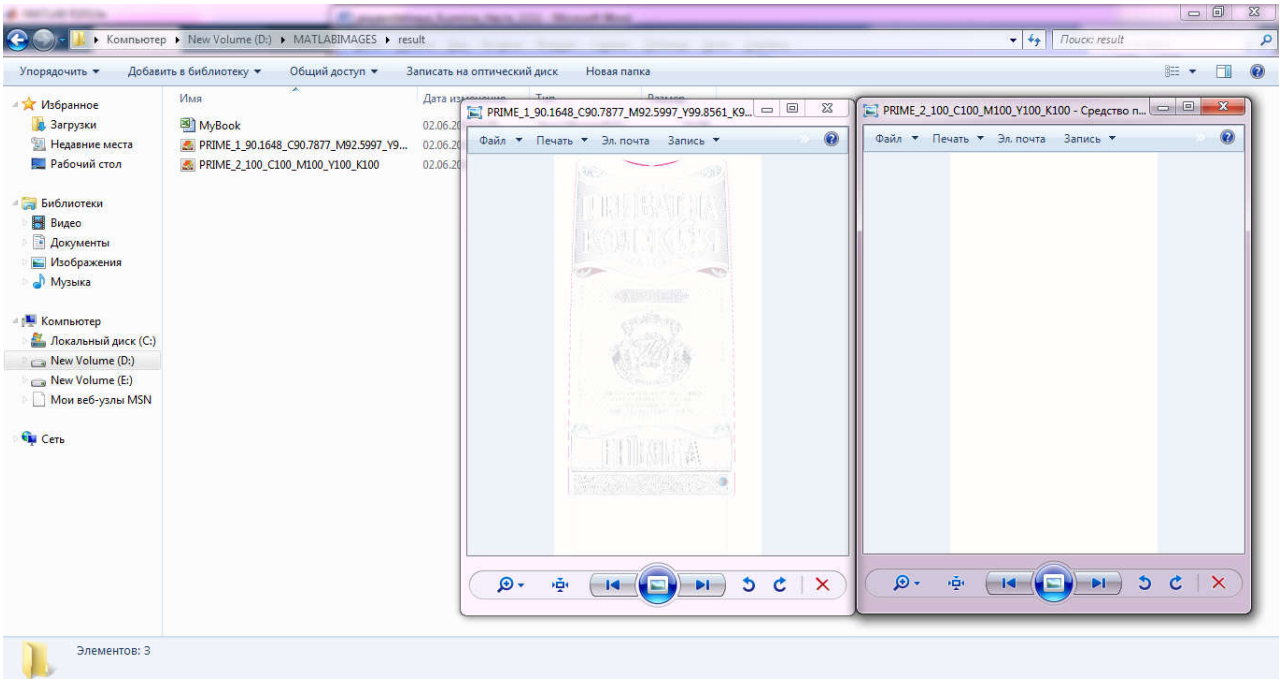


Рисунок 7 – Вміст теки result

Результуючі зображення мають унікальні імена. Формат запису імені можна представити таким чином: Ім'я зображення_для перевірки, відсоток абсолютного збігу_відсоток збігу по блакитному каналу_ відсоток збігу по пурпурному каналу_ відсоток збігу по жовтому каналу_ відсоток збігу по чорному каналу.

Для зручності аналізу інформації дані кожного результуючого зображення вносяться в таблицю Excel. Загальний формалізований вигляд таблиці представлений в таблиці 1.

Таблиця 1 – Загальний вигляд таблиці Excel

	ABS	C	M	Y	K	result
Ім'я результату	1 – 100%	1 – 100%	1 – 100%	1 – 100%	1 – 100%	+ або -
Ім'я результату	1 – 100%	1 – 100%	1 – 100%	1 – 100%	1 – 100%	+ або -

Осередки ABS, C, M, Y, K означають збіг абсолютний, по блакитному, пурпурному, жовтому і чорному каналах відповідно. У стовпці result позначкою + позначається ідентичність порівнюваних зображень, позначкою – неспівпадіння.

Розглянемо порівняння двох копій з одним оригінальним зображенням і результати обробки зображень програмою. Оригінальне зображення для перевірки і результат представлені на рис. 8 та 9 відповідно.

Друге результуюче зображення обведене в рамку, оскільки воно є результатом абсолютного збігу порівнюваної пари, тобто повністю біле.

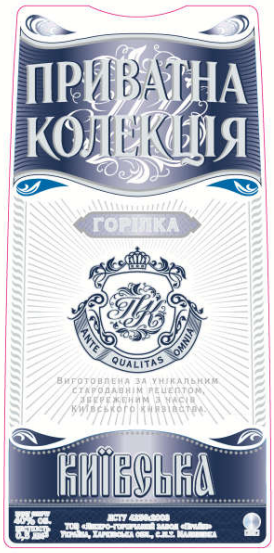


Рисунок 8 – Оригінал

Рисунок 9 – зображення для перевірки (два ліворуч) та і результати (два праворуч)

Приклад таблиці Excel для приведених зображень представлений в табл. 2.

Таблиця 2 – Результат порівняння

	ABS	C	M	Y	K	result
PRIME_1	90,1647	90,79	92,599	99,856	91,87	+
PRIME_2	100	100	100	100	100	-

Таким чином ці таблиці підтверджують неспівпадіння першого зображення для перевірки та повний збіг другого. Оскільки перше зображення не співпало, необхідно дослідити його захисні елементи ретельніше.

2.2 Дослідження порівняння етикетки із захисними елементами

При отриманні результатів виконання програми необхідно розуміти, де саме проявляються неспівпадіння. Візуальне відображення частково вирішує цю проблему. Проте в процесі контролю захисних елементів на стадії додрукарської підготовки іноді доцільно перевіряти окремі частини макету, що містять захисні елементи. Наприклад, використання нанесення лаку на певну область, що

відображено на зображенні оригінал-макету в окремому шарі, перекриває інші захисні елементи. Також необхідно враховувати, що захисні елементи можуть займати невелику площу поверхні зображення, і практично повний збіг зображень не гарантує того, що малий відсоток неспівпадіння не буде відноситься саме до цих елементів.

Отже, бувають випадки, коли фрагментарна перевірка зображень потрібна. Для дослідження дії перевірки фрагментів зображення розглянуто етикетку, представлену на рисунку 10.

Рисунок 10 – Етикетка із захисними елементами

Ця етикетка має наступні захисні елементи:

– фігурне висікання (1);
– OVI-фарба (2);
– лакування (3);
– використання pantone (4);
– антисканерна сітка (5) ;
– псевдоірисовий друк (6);
– текст, тиснення фольгою (7);
– захисна пірнаюча флуоресцентна металізована нитка (8);
– мікротекст (9);
– гілйоширна композиція із змінною товщиною штриха (10);
– приховане зображення у вигляді гілйоширної розетки (11).

Усі використані види захисних елементів застосовуються на різних етапах поліграфічного процесу, проте задаються на додрукарській стадії. Така комбінація істотно підвищує рівень захисту етикетки. Тому, безумовно, передусім необхідно контролювати правильність відтворення усіх елементів захисту.

Існує можливість програмно розділити зображення на фрагменти, що містять елементи захисту. Проте це не є завданням дослідження, тому для розбиття зображення використовуються шари програми Adobe Illustrator, за

допомогою якої був створений оригінал-макет етикетки. Аналогічні дії виконуються для зображення для перевірки. Необхідно стежити за тим, щоб відповідні фрагменти отримували коректні імена.

Порівняння форми висікання показало, що зображення для перевірки має відмінності, що виражаються в невідповідності форми. На рисунку 1 зліва направо відображені форми висікання оригінального зображення, зображення для перевірки а також результуючого зображення.

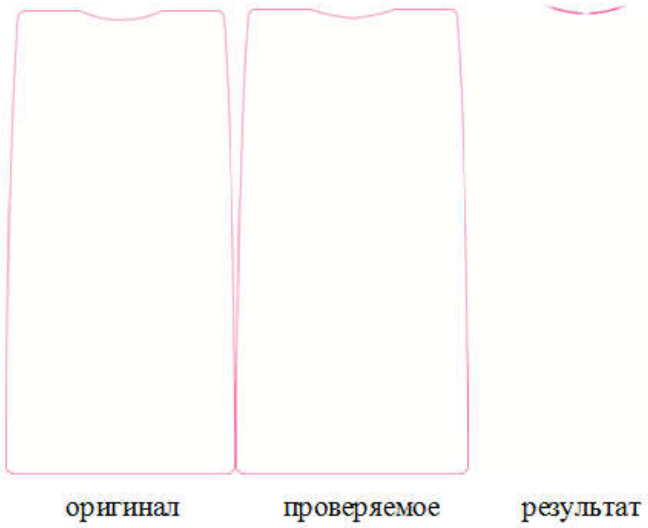


Рисунок 11 – Порівняння форми висікання

Кольоромінливі фарби OVI мають металевий відблиск та ефект переливання, який виходить в результаті зломлення променів, також важливий колір основи, на яку наноситься ця фарба. Для кольорів основи віддається перевага темним відтінкам. Сама фарба може мати різні відтінки, звичайно це жовто-зелені або темно-блакитні кольори [9]. Результат перевірки області з нанесенням OVI-фарби на ідентичність оригіналу, представлений на рис. 12.

Одним з важливих компонентів, що використовуються на післядрукарській стадії, але задаються на додрукарській, є лакування. Існують безліч лаків, що мають різні властивості, тому дуже важливо чітко визначати де і який лак повинен наноситися. Також слід зазначити, що окрім лаку існують і інші захисні покриття, проте принцип їх позначення аналогічний принципу нанесення лаку. На рисунку 13 показано порівняння зон нанесення лаку.

Рисунок 12 – Порівняння основ нанесення фарби OVI

Рисунок 13 – Порівняння лакування

Використання певних фарб з каталогів кольорів PANTONE гарантує отримання однакового кольору при друці на усіх відбитках. Тому застосування таких фарб особливе актуально для етикетково-пакувальної продукції.

Помилка при виборі кольору викликає брак віддруковуваного накладу. Застосування кольорів, сформованих на підставі моделі СМУК, також дає відмінності від кольорів систем кольору PANTONE кольорів. На рисунку 14 продемонстровано порівняння фрагментів, які мають бути віддруковані такими фарбами.

Порівняння антисканерних сіток можливе тільки за наявності оригіналу і зображення для перевірки в цифровому виді. На цьому прикладі антисканерна сітка виконується за допомогою псевдоірисового друку, тобто із застосуванням градієнту. Виконання перевірки фрагмента, що містить цю антисканерну сітку , представлене на рисунку 15. Рисунок показує, що антисканерна сітка на зображенні для перевірки виконана одним тоном, а не градієнтом. Для більшої наочності на рисунку збільшений контраст.

Рисунок 14 – Порівняння pantone

Рисунок 15 – Порівняння сітки антисканера і псевдоірисового друку

Використання кольорового тиснення значно збільшує рівень захисту етикетки. Порівняння елементів, що виконуються тисненням, приведене на рисунку 16.

Рисунок 16 – Порівняння елементів, що виконуються кольоровим тисненням

Захисна нитка, впроваджена в оригінальному зображенні, в результаті перевірки також має відмінності. Застосування захисної пірнаючої нитки відображено на рисунку 17.

Рисунок 17 – Порівняння пірнаючої захисної нитки

Відмінності фрагмента, що містить мікротекст, свідчать про фальсифікацію зображення для перевірки. В даному випадку замість тексту використана тонка смуга, що імітує мікротекст. Порівняння представлене на рисунку 18.

Рисунок 18 – Порівняння мікротексту

У нижній частині даної етикетки застосовано гільйоширну композиція із змінною товщиною штриха і змінним кольором.

На рис. 19 продемонстрована оригінальна композиція для перевірки і результуюче зображення.

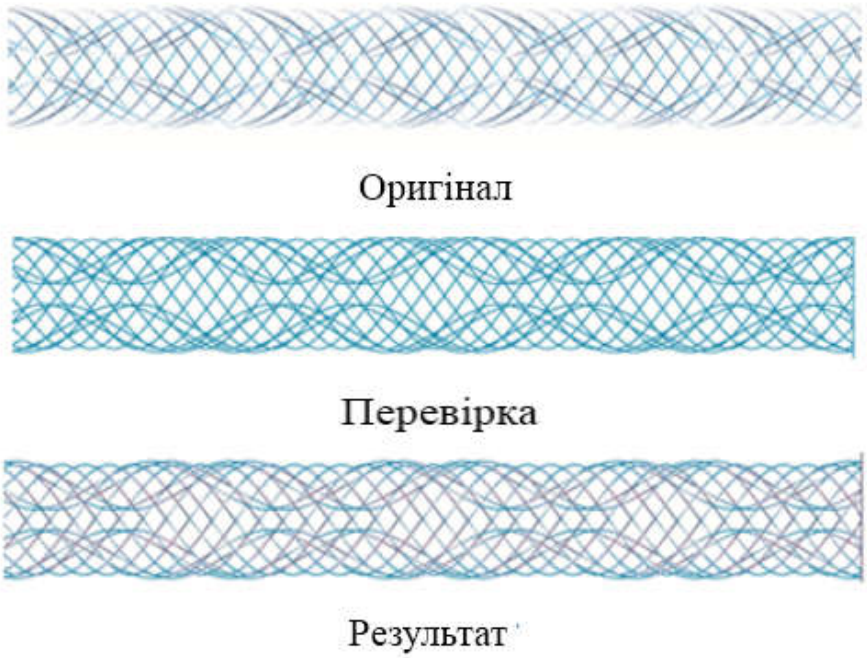


Рисунок 19 – Порівняння мікротексту

Приховане зображення у вигляді гильоширної розетки, проявляється при розгляді етикетки в косопадаючих променях світла. На рисунку 20 зображені оригінальна, така для перевірки і результуюча розетки.

Рисунок 20 – Порівняння мікротексту

Таким чином, детальний фрагментарний аналіз дозволяє побачити неспівпадіння різних захисних елементів, використовуваних в етикетково-пакувальної продукції. Результати виконання програми наочно представляють ці неспівпадіння.

2.3 Аналіз ефективності роботи програми

Для оцінки ефективності автоматичної дії скрипта необхідно виконати перевірку роботи з різною кількістю оригінальних зображень для перевірки.

Приклад дії створеного скрипта для однієї пари порівнюваних зображень наведений на рисунках 21 та 22, що демонструють оригінальну етикетку, а також зображення для перевірки та результуюче зображення. Підсумковим файлом є зображення, отримане в результаті віднімання, у форматі tif. Кожному збереженому файлу привласнюється унікальне ім’я, яке містить назву зображення для перевірки, абсолютний збіг з оригіналом у відсотках, поканальний збіг у відсотках (окремо виділений збіг по кожній колірній координаті CMYK).

Рисунок 21 – Порівнювані зображення

Рисунок 22 – Результуюче зображення

Дані обчислень підтверджують візуально виявлені відхилення в кольорах. Збіг даного зразка склав 94,6 % по блакитному і чорному каналам, 89,4 % – по пурпурному і жовтому. Абсолютний збіг зображень рівний 89,4 % що пояснюється збігом декількох кольорів в певній кількості пікселів.

Швидкість поетапного виконання порівняння представлених зразків отримана за допомогою profiler приведена в таблиці 3. Підсумкова швидкість обробки цієї пари зображень склала 9, 746 с. Швидкість обробки конкретного файлу залежить від його роздільної здатності та займаного ним об'єму пам'яті.

За допомогою Adobe Photoshop можна зробити виміри колірних значень пікселів інструментом піпетка. Отже, застосувавши формули, використані в програмному коді, можна вручну визначити абсолютний та поканальний збіг.

Таблиця 3 – Швидкість поетапного виконання програми

Function Name	Calls	Total Time, з	Self Time, з	Total Time Plot (dark = self time)
1	2	3	4	5
Untitled	1	9.993	9.746	
fileparts	2	0.004	0.004	
fullfile	2	0.005	0.002	
fullfile>addTrailingFileSep	2	0.001	0.001	
fullfile>refinePath	2	0.001	0.001	
imabsdiff	1	0.013	0.002	
images\private\checkForSame SizeAndClass	1	0.001	0.001	
images\private\imabsdiffmex ((MEX – file)	1	0.010	0.010	
imagesci\private\imftype	2	0.005	0.001	
imagesci\private\istif	2	0.002	0.002	
imagesci\private\readtif	2	0.084	0.001	
imagesci\private\readtif>check_info	2	0.000	0.000	
imagesci\private\readtif>parse_args	2	0.001	0.001	
imagesci\private\rtifc (MEX – file)	2	0.081	0.081	
imagesci\private\writetif	1	0.094	0.006	
images\writetif>parse_param_value_pairs	1	0.005	0.002	
imagesci\private\wtifc (MEX – file)	1	0.084	0.084	

Продовження таблиці 3

1	2	3	4	5
imformats	4	0.004	0.001	
imformats>find_in_registry	4	0.004	0.004	
imread	2	0.093	0.004	
imread>parse_inputs	2	0.001	0.001	
imwrite	1	0.106	0.003	
imwrite>get_format_from_filename	1	0.003	0.001	
imwrite>parse_inputs	1	0.003	0.003	
imwrite>validateSizes	1	0.002	0.001	
intmax	1	0.001	0.001	
iscellstr	3	0.001	0.001	
ispc	4	0.000	0.000	
num2str	5	0.007	0.004	
num2str>convertUsingRecycledSprintf	5	0.002	0.002	
num2str>handleNumericPrecision	5	0.002	0.000	
strcat	4	0.020	0.019	
validatestring	3	0.002	0.000	
validatestring>checkInputs	3	0.001	0.001	

Припустимо, що є два пікселі, які співпадають тільки по одному каналу. Для спрощення обчислень поріг чутливості прийнятий рівним нулю. Виконаємо розрахунки для пікселів з координатами: 100С, 60М, 60У, 84К та 98С, 45М, 44У, 84К. Колірні координати за результатами віднімання дорівнюють 2С, 15М, 16У, 0К. Таким чином, збіг по чорному кольору складає 100%, абсолютний та CMY збіги відсутні. Час виконання цієї операції вручну складає 75 секунд. Віднімання і збереження зображень інструментами Photoshop не виконувалося, тому час на ці дії враховується, проте слід розуміти, що час порівняння вручну вище 75 секунд. При цьому час виконання усіх дій, з урахуванням віднімання, збереження зображень і супутніх цьому функцій для скрипта займає 0,105 секунд. Очевидно, що код виконує дії набагато швидше. Матриця аналізованого зображення має розміри 1279x888 пікселів. Це означає, що час виконання віднімання вручну більше як мінімум в 1135752 рази і займе не менше 986 діб. Таким чином, повне виконання аналізу двох зображень вручну абсолютно неможливе.

Дослідження пакетної обробки зображень включає порівняння різної кількості оригіналів і копій. Ці обробки різної кількості зображень приведені в таблиці 4.

Таблиця 4 – Час обробки різної кількості зображень

Кількість зображень		Час обробки, з	
оригіналів	що перевіряються	загальне	для однієї пари
1	1	9, 746	9, 746
1	5	59.418	11,8836
5	5	28.939	5,7878
10	10	76.242	7,6242
20	35	316.418	9,04

Таким чином, середній час порівняння пари зображень дорівнює 7,008 з, що підтверджує швидкодію програми. Швидкість виконання в основному залежить від пам’яті, займаної порівнюваними зображеннями.

Висновки

Розроблений програмний засіб дає можливість порівнювати зразки етикетково-пакувальної продукції. Рекомендується проводити порівняння з нульовим пороговим значенням чутливості програми до відмінностей. Передбачена можливість коригувати це значення для компенсації різних погрішностей. Проте якщо в цьому немає необхідності, уся робота програми складається із запуску і отримання результату на виході. Природно, що порівнювані зображення мають бути підготовлені і розташовані в теках, відведених для оригінальних зображень для перевірки. Зокрема, це стосується фрагментарної перевірки, при якій відбувається розбиття на частини, що містять захисні елементи.

Результати дослідження різних зразків етикетково-пакувальної продукції, що містять захисні елементи, показали, що створений програмний засіб надає очікувану інформацію. Виконання коду повністю відповідає алгоритму. Збереження даних в таблиці Excel дозволяє порівнювати результати дослідження різних пар зразків і навіть вести статистику збігів. Так в ході дослідження були виявлені як співпадаючі, так і неспівпадаючі зразки. З тридцяти п’яти перевірених пар співпали лише чотири зображення, що підтверджує актуальність проведення подібної перевірки.

Результуюча таблиця отримана швидко для аналізованого об’єму пам’яті та кількості зображень (за 316 секунд) і інформативна. Показники збігу зображень по каналах дають корисну інформацію про колірні відхилення зображення для перевірки. По отриманих результатах аналітичного і візуального порівняння можна досить точно визначати причини виникнення неспівпадінь і при необхідності їх усувати.

Слід зазначити, що порівняння різних елементів захисту також відбувається по-різному через міру їх машиночитабельності. Так колірна різниця буде відображена наочніше, ніж незначні відмінності в захисних елементах типу гильйошей, тангірних сіток, мікрозображень і інших елементах.

Проте на підставі цих даних можна зробити висновки про доцільність застосування елементу захисту в конкретному зразку продукції, а також про його ефективність.

Так, наприклад, мікротекст, при порівнянні на зображеннях з роздільною здатністю 300 dpi, залишається читабельним аж до розміру 0,4 pt, при меншому розмірі текст стає невиразний, проте візуально відмінність проявляється. Звідси витікає, що мікротекст розміром менше 0,4 pt є ефективним елементів захисту, але при цьому вимагає додаткових методів перевірки.

Тобто наявність цього елементу передбачає його перевірку в умовах професійного оточення і закритого (неоголошеного) захисту.

В ході дослідження з'ясовано, що при збільшенні роздільної здатності порівнюваних зображень результати є точнішими. Рекомендована роздільна здатність для порівнюваних зображень – 300 dpi.

Також значення абсолютного збігу може показати ефективність застосування комбінованого захисту у порівнянні з аналогічним зразком продукції, що не має такого захисту.

Список літератури

1. Коншин А.А. Защита полиграфической продукции от фальсификации. М.: Синус, 1999. 160 с.

2. Рынок упаковки. Состояние, тренды и инновации // Полимерные материалы. 2021. № 11. С. 43-50.

3. Global packaging industry. URL: https://www.smitherspira.com/market-reports/global-packaging-industry-expected-to-reach-820-billion-by-2016.aspx.

4. Шарифуллин М. Сага об этикетке // Sales business. 2007. №1. С. 12-15.

5. Защищенная полиграфическая продукция // КомпьюАрт. URL: http://www.compuart.ru/article.aspx?id=9303&iid=393/.

6. Шарифуллин М. Защита прежде всего // Publish. 2000. №7. С. 22-24.

7. Полиграфические методы защиты // КомпьюАрт. URL: http://www.compuart.ru/article.aspx?id=8348&iid=336/.

8. Способы защиты документов // Бюро научно-технической информации. URL: http://www.bnti.ru/showart.asp?aid=940&lvl=01.03.05.

9. Киричок П.О., Коростіль Ю.М., Шевчук А.В. Методи захисту цінних паперів та документів суворого обліку. К.: НТУУ "КПІ", 2008. 368 с.

10. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. Введ. 1992-01-01. М. : Изд-во стандартов, 1992. 24 с.

11. Дьяконов В.П. MATLAB. Полный самоучитель. М.: ДМК Пресс, 2012. 768 с.

12. Бизюк А.В., Жернова П.Е. Расчет обобщенного показателя защищённого полиграфического изделия для информационной системы // Бионика интеллекта. 2016. №1 (86). С. 63-67.

13. Бизюк А.В., Шамо И.И. Решение задачи пространственного сегментирования с применением методов интеллектуального анализа данных // Информационные системы и технологии: материалы 2-й Международ. науч.-техн. конф. (16-22 сентября 2013, Евпатория-Харьков). 2013. С. 134-135.

14. Кузьмина К.В., А.В. Бизюк Классификация методов защиты полиграфической продукции // Информационные системы и технологии: материалы 3-й Междунар. науч.- техн. конф. ИСТ-2014 (15-21 сент. 2014 г., Харьков). 2014. С. 206-207.

15. Жернова П.Е., Бизюк А.В. Методы оценки интегрального показателя для защищенного полиграфического изделия // Полиграфические, мультимедийные и WEB-технологии (PMW-2016): тез. докл. 1-й Междунар. науч.-техн. конф. (16-20 мая 2016 г., Харьков). 2016. Т. 1. С. 47-48.

16. Жернова П.Е., Бизюк А.В. Оптимизация выбора полиграфической защиты для упаковочно-этикеточной продукции // Информационные системы и технологии: материалы 2-й Международ. науч.-техн. конф. (16-22 сентября 2013 г., Евпатория-Харьков). 2013. С. 142-143.